# **Lab Manual - OOP**

# **Operators Overloading and Friend Functions**

## Objectives

The purpose of this lab is to explain and implements all types of operators that can be overload and to provide an overview of Friend Functions.

## Sample Tasks

**Assignment Operator Overloading**

**Code:**

|  |
| --- |
| #include<iostream>  using namespace std;  class abc  {  public:  abc()  {  i=0;  }  abc(int ii)  {  i=ii;  }  void operator = (abc obj)  {  i=obj.i;  cout<<"Assignment Opeartor has been overloaded"<<endl;  }  void print()  {  cout<<"i = "<<i<<endl;  }  private:  int i;  };  int main()  {  abc a(10);  abc b(100);  b=a;  a.print();  b.print();  return 0;  } |

**Sample code**

**Binary Operators Overloading Plus Operator Overloading**

**Code:**

|  |
| --- |
| #include<iostream>  using namespace std;  class abc  {  public:  abc()  {  i=0;  }  abc(int ii)  {  i=ii;  }  abc operator + (abc obj)  {  abc a;  a.i=i+obj.i;  cout<<"Plus Opeartor has been overloaded"<<endl;  return a;  }  void print()  {  cout<<"i = "<<i<<endl;  }  private:  int i;  };  int main()  {  abc a(10);  abc b(100);  abc c;  c=a+b;  a.print();  b.print();  c.print();  return 0;  } |

**Sample code**

**Overload - .\* , and / operators and show their results.**

**Less than Operator Overloading**

**Code:**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  class abc  {  public:  abc()  {  i=0;  }  abc(int ii)  {  i=ii;  }  bool operator < (abc obj)  {  cout<<"Less than Opeartor has been overloaded"<<endl;  return i < obj.i;  }  void print()  {  cout<<"i = "<<i<<endl;  }  private:  int i;  };  int main()  {  abc a(10);  abc b(100);  cout<< (a<b)<<endl;  cout<< (b<a)<<endl;  return 0;  } |

**Sample Code**

**Overload Operator <= ,> , > = , = = , != , and show results.**

**Unary Operator Overloading**

**Code:**

|  |
| --- |
| #include<iostream>  using namespace std;  class abc  {  public:  abc()  {  i=0;  }  abc(int ii)  {  i=ii;  }  abc operator ++ ()  {  abc temp;  temp.i = i++;  cout<<"Increment Opeartor has been overloaded"<<endl;  return temp; // returns current value of i  }  void print()  {  cout<<"i = "<<i<<endl;  }  private:  int i;  };  int main()  {  abc a(10);  a.print();  ++a;  return 0;  } |

**Sample Code**

**Overload Decrement -- Operator and show results.**

**Insertion (>>) and Extraction (<<) operators Overloading**

**Code:**

|  |
| --- |
| #include<iostream>  using namespace std;  class abc  {  public:  abc()  {  i=0;  }  abc(int ii)  {  i=ii;  }  friend ostream &operator<<( ostream &output, const abc &obj );  friend istream &operator>>( istream &input, abc &obj );  private:  int i;  };    ostream &operator<<( ostream &output, const abc &obj )  {  output<< "i = "<<obj.i<<endl;  output<<"Extraction Operator has been overloaded.";  return output;  }  istream &operator>>( istream &input, abc &obj )  {  input>> obj.i;  return input;  }  int main()  {  abc a;  cout<< "Enter Value if i:\t";  cin>> a;  cout<<a;  return 0;  } |

**Friend Functions**

**Friend Functions:**

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.

A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

To declare a function as a friend of a class, precede the function prototype in the class definition with keyword friend as follows:

**class Box**

**{**

**double width;**

**public:**

**double length;**

**friend void printWidth( Box box ); // Declaration of Friend Function**

**void setWidth( double wid );**

**};**

To declare all member functions of class ClassTwo as friends of class ClassOne, place a following declaration in the definition of class ClassOne:

friend class ClassTwo;

Consider the following sample task.

**Sample Code :**

|  |
| --- |
| #include <iostream>  using namespace std;  class Box  {  double width;  public:  friend void printWidth( Box box );  void setWidth( double wid );  };  // Member function definition  void Box::setWidth( double wid )  {  width = wid;  }  // Note: printWidth() is not a member function of any class.  void printWidth( Box box )  {  /\* Because printWidth() is a friend of Box, it can  directly access any member of this class \*/  cout<< "Width of box : " << box.width <<endl;  }  // Main function for the program  int main( )  {  Box box;  // set box width without member function  box.setWidth(10.0);  // Use friend function to print the wdith.  printWidth( box );  return 0;  } |

## Lab Tasks

**Task 1:**

(RationalNumber Class) Create a class RationalNumber (fractions) with the following capabilities:

1. Create a constructor that prevents a 0 denominator in a fraction, reduces or simplifies fractions that are not in reduced form and avoids negative denominators.
2. Overload the addition, subtraction, multiplication and division operators for this class.
3. Overload the relational and equality operators for this class.

**Task 2:**

(Polynomial Class) Develop class Polynomial. The internal representation of a Polynomial is an array of terms. Each term contains a coefficient and an exponent, e.g., the term 2x4 has the coefficient 2 and the exponent 4. Develop a complete class containing proper constructor and destructor functions as well as set and get functions. Assume that the polynomial is in one variable and maximum possible degree is 5. The class should also provide the following overloaded operator capabilities:

1. Overload the addition operator (+) to add two Polynomials.
2. Overload the subtraction operator (-) to subtract two Polynomials.
3. Overload the assignment operator to assign one Polynomial to another.
4. Overload the multiplication operator (\*) to multiply two Polynomials.
5. Overloaded insertion and extraction operator (>>, <<)

**Task 3:**

Write a program with a class that contains an array of integers. Initialize the integer array in the constructor of the class. Then create two friend functions to the class to find the largest and smallest integers in the array.

Create a destructor that sets all of the elements in the array to 0.